Se instala el jdk desde este link  
<http://www.oracle.com/technetwork/es/java/javase/downloads/jdk7-downloads-1880260.html>

En mi caso instale el Windows i586, mi pc es de 32 bits.

Instalando Maven

<http://maven.apache.org/download.cgi#Installation>



Se instala la última versión de maven

Y se siguen los pasos para instalar en Windows

(en este caso no se ejecuto la acción opcional)

Al momento de verificar en los símbolos de sistema que la instalación se llevo a cabo con éxito es necesario correr los símbolos de sistema como administrador y además navegar hasta el bin de la carpeta donde esta maven

Plugins:

Instalar maven utilities

Instalar spring web mvc

Build with dependencies (descarga las dependencias)

Seguir este paso:

<http://projects.spring.io/spring-security/#quick-start>

adjuntando ese par de dependencias

en el proyecto-> Project files -> POM

**Configuración del CAS:**

1. Descargar el CAS (Versión 3.5.2.1 - zip) desde:

<http://www.jasig.org/cas_server_3_5_2_1_release>

1. Abrir el proyecto Web del cas, y en el archivo “web.xml” agregar lo siguiente:

<filter>

<filter-name>CAS Authentication Filter</filter-name>

<filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>

<init-param>

<param-name>casServerLoginUrl</param-name>

<param-value>https://auth.espol.edu.ec/login</param-value>

</init-param>

<init-param>

<param-name>service</param-name>

<param-value>http://localhost:5000/rongo</param-value>

</init-param>

</filter>

<filter>

<filter-name>CAS Validation Filter</filter-name>

<filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>

<init-param>

<param-name>casServerUrlPrefix</param-name>

<param-value>https://auth.espol.edu.ec/</param-value>

</init-param>

<init-param>

<param-name>service</param-name>

<param-value>http://localhost:5000/rongo</param-value>

</init-param>

</filter>

<filter>

<filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>

<filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>CAS Authentication Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>CAS Validation Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

1. Compilar el proyecto con las dependencias.
2. Generar un certificado para permitir una conexión “segura”. Véase “**Step 2: Used keytool to self-author a server certificate for DEMO**” en el siguiente enlace: <https://wiki.jasig.org/display/CASUM/Demo> Importante: El certificado es únicamente para pruebas locales, no debe ser subido a un servidor público.
3. Configurar un conector SSL para el servidor en donde será desplegado el archivo .war del CAS. En este caso se usa “apache-tomcat-7.0.54”. Se edita el archivo “server.xml”

<Connector port="5001" protocol="org.apache.coyote.http11.Http11Protocol"

maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

clientAuth="false" sslProtocol="TLS" keystorePass="changeit"

truststoreFile="C:\Program Files\Java\jdk1.7.0\_07\jre\lib\security\cacerts"/>

1. Iniciar Tomcat.
2. Desplegar “CAS.war”
3. Compilar Rongo-web, y desplegar.
4. Acceder a <https://localhost:port> y probar el CAS.

Como solucionar el problema del Handshake?:

-Abrir el cmd y navegar hasta la carpeta InstallCacert provista en el proyecto

-Ejecutar el programa InstallCacert de la carpeta InstallCacert con el parámetro auth.espol.edu.ec-1

-se creara un archivo llamado jssecacerts

-Viajar hasta la carpeta de nuestro servidor, en netbeans ir a la pestaña servicios -> servidores-> Glassfish, hacer click derecho y hacer click en propiedades, allí debería estar la ruta de glassfish

-Ir hasta dicha ruta y abrir domain🡪 domain1🡪congif y reemplazar cacerts.jks con jssecacerts (mantener una versión antigua de cacerts.jks por si acaso)

Como añadir spring al proyecto:

Hacer click derecho al proyecto e ir a propiedades

-Buscar Frameworks y añadir spring. A continuación se crearan un par de archivos:

Para configurar cas se debe añadir esto al archivo application-context:

<sec:authentication-manager alias="authenticationManager">

<sec:authentication-provider ref="preAuthAuthProvider"/>

</sec:authentication-manager>

<bean id="springSecurityFilterChain" class="org.springframework.security.web.FilterChainProxy">

<sec:filter-chain-map path-type="ant">

<sec:filter-chain pattern="/" filters="casValidationFilter, wrappingFilter" />

<sec:filter-chain pattern="/secure/receptor" filters="casValidationFilter" />

<sec:filter-chain pattern="/j\_spring\_security\_logout" filters="logoutFilter,etf,fsi" />

<sec:filter-chain pattern="/\*\*" filters="casAuthenticationFilter, casValidationFilter, wrappingFilter, sif,j2eePreAuthFilter,logoutFilter,etf,fsi"/>

</sec:filter-chain-map>

</bean>

<bean id="sif" class="org.springframework.security.web.context.SecurityContextPersistenceFilter"/>

<bean class="org.jasig.cas.client.validation.Cas20ServiceTicketValidator" id="ticketValidator">

<constructor-arg index="0" value="https://auth.espol.edu.ec/" />

<property name="proxyGrantingTicketStorage" ref="proxyGrantingTicketStorage" />

<property name="proxyCallbackUrl" value="https://auth.espol.edu.ec/secure/receptor" />

</bean>

<bean id="logoutFilter" class="org.springframework.security.web.authentication.logout.LogoutFilter">

<constructor-arg value="/"/>

<constructor-arg>

<list>

<bean class="org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler"/>

</list>

</constructor-arg>

</bean>

<bean id="fsi" class="org.springframework.security.web.access.intercept.FilterSecurityInterceptor">

<property name="authenticationManager" ref="authenticationManager"/>

<property name="accessDecisionManager" ref="httpRequestAccessDecisionManager"/>

<property name="securityMetadataSource">

<sec:filter-invocation-definition-source>

<!--

<sec:intercept-url pattern="/secure/extreme/\*\*" access="ROLE\_SUPERVISOR"/>

<sec:intercept-url pattern="/secure/\*\*" access="ROLE\_USER"/>

<sec:intercept-url pattern="/\*\*" access="ROLE\_USER"/>

<sec:intercept-url pattern="/logout.jsp" access="ROLE\_ANONYMOUS"/>

<sec:intercept-url pattern="/index2.html" access="ROLE\_SUPERVISOR"/>

-->

<sec:intercept-url pattern="/\*\*" access="ROLE\_USER"/>

</sec:filter-invocation-definition-source>

</property>

</bean>

<bean id="proxyGrantingTicketStorage" class="org.jasig.cas.client.proxy.ProxyGrantingTicketStorageImpl" />

<bean id="etf" class="org.springframework.security.web.access.ExceptionTranslationFilter">

<property name="authenticationEntryPoint" ref="preAuthEntryPoint"/>

</bean>

<bean id="casAuthenticationFilter" class="org.jasig.cas.client.authentication.AuthenticationFilter">

<property name="casServerLoginUrl" value="https://auth.espol.edu.ec/login" />

<property name="serverName" value="https://localhost:8181"/>

</bean>

<bean id="casValidationFilter" class="org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter">

<property name="serverName" value="https://localhost:8181" />

<property name="exceptionOnValidationFailure" value="true" />

<property name="proxyGrantingTicketStorage" ref="proxyGrantingTicketStorage" />

<property name="redirectAfterValidation" value="true" />

<property name="ticketValidator" ref="ticketValidator" />

<property name="proxyReceptorUrl" value="/secure/receptor" />

</bean>

<bean id="preAuthAuthProvider" class="org.springframework.security.web.authentication.preauth.PreAuthenticatedAuthenticationProvider">

<property name="preAuthenticatedUserDetailsService">

<bean id="userDetailsServiceWrapper" class="org.springframework.security.core.userdetails.UserDetailsByNameServiceWrapper">

<property name="userDetailsService" ref="userService"/>

</bean>

</property>

</bean>

<bean id="roleVoter" class="org.springframework.security.access.vote.RoleVoter"/>

<bean id="httpRequestAccessDecisionManager" class="org.springframework.security.access.vote.AffirmativeBased">

<property name="allowIfAllAbstainDecisions" value="false"/>

<property name="decisionVoters">

<list>

<ref bean="roleVoter"/>

</list>

</property>

</bean>

<sec:user-service properties="WEB-INF/users.properties" id="userService"></sec:user-service>

<bean id="j2eePreAuthFilter" class="org.springframework.security.web.authentication.preauth.j2ee.J2eePreAuthenticatedProcessingFilter">

<property name="authenticationManager" ref="authenticationManager"/>

<property name="authenticationDetailsSource">

<bean class="org.springframework.security.web.authentication.WebAuthenticationDetailsSource" />

</property>

</bean>

<bean id="preAuthEntryPoint" class="org.springframework.security.web.authentication.Http403ForbiddenEntryPoint" />

<bean id="wrappingFilter" class="org.jasig.cas.client.util.HttpServletRequestWrapperFilter" />

</beans>

-Para lo cual se va a necesitar el prefijo security:

xmlns:sec=<http://www.springframework.org/schema/security>

en squema location:

http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.0.xsd

además se deberán añadir ciertas dependencias al POM:

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-web</artifactId>

<version>3.0.7.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-config</artifactId>

<version>3.0.7.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-cas</artifactId>

<version>3.1.0.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-core</artifactId>

<version>3.0.7.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-openid</artifactId>

<version>3.0.7.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>3.0.3.RELEASE</version>

<scope>runtime</scope>

</dependency>

(la dependencia spring-security-cas posiblemente ya esta en la versión anterior)

Por ultimo añadir el archivo user.properties con información de los roles de cada usuario

La manera de asignar roles a cada usuario es:  
usuario=random,ROLE\_ADMIN,enabled

En este caso usamos el valor random como contraseña porque tenemos en cambio otro sistema de autenticación distinto al que nos asignara los roles, asi que no importa la contraseña que se le especifique al usuario

En vez de poner ROLE\_ADMIN solamente a un usuario se le pueden añadir cualquier numero de roles, podría por ejemplo esa línea ser:

usuario=random,ROLE\_ADMIN,ROLE\_USER,ROLE\_AYUDANTE,enabled

Ejecutar build with dependencies y resolver todas las dependencias

Como manejar roles:

Para agregar roles

Los accesos a las paginas se manejan en el Bean con id: fsi

<sec:intercept-url pattern="/\*\*" access="ROLE\_USER"/>

En dicho pattern lo que se indica es que en cada link que vaya despues de “/” se podrá entrar solo quien tenga el acceso de ROLE\_USER

Si queremos por ejemplo proteger una pagina que se llame mirol.html para que solo pueda ser accedido por usuarios de rol = ROLE\_MIROL habrá que añadir esta línea

<sec:intercept-url pattern="/mirol.html" access="ROLE\_MIROL"/>

Esto sobreescribira las condiciones para dicha pagina solamente donde mirol.html solo podra ser accedido por un usuario de acceso ROLE\_MIROL

Si por otro lado queremos que toda una carpeta (por ejemplo una carpeta llamada “secured”) sea protegida podemos usar:

<sec:intercept-url pattern="/secured/\*\*" access="ROLE\_MIROL"/>